Clase `BufferManager`

- Funcionalidad: Administra la interacción entre el disco físico y el buffer pool. Coordina la carga de páginas, estadísticas y control de uso.
- Métodos:
    - `BufferManager(int num_frames, DiscoFisico* mydisk)`
        - Objetivo: Inicializar el buffer pool y enlazar el disco.
        - Input: número de marcos, puntero al disco físico.
        - Output: ninguno.

    - `string* acceder(int id_bloque, Operacion op)`
        - Objetivo: Acceder o cargar una página y devolver su contenido.
        - Input: ID de bloque, operación (Leer, Insertar, Eliminar).
        - Output: puntero a string (contenido).

    - `void ver_tabla()`
        - Objetivo: Mostrar estado del buffer pool.
        - Input: ninguno.
        - Output: impresión por consola.

    - `void high_dirty_bit(int id)`
        - Objetivo: Marcar página como modificada.
        - Input: ID de página.
        - Output: ninguno.

    - `void pin(int id)` / `void unpin(int id)`
        - Objetivo: Marcar/desmarcar uso de un marco.
        - Input: ID de marco.
        - Output: ninguno.

    - `void guardar(int id)`
        - Objetivo: Guardar cambios al disco si el marco está sucio.
        - Input: ID de página.
        - Output: impresión por consola.

    - `void eliminar(int id)`
        - Objetivo: Eliminar página del buffer, guardando si es necesario.
        - Input: ID de página.
        - Output: ninguno.

Clase `BufferPool`

- Funcionalidad: Maneja los marcos de memoria, la política LRU, estadísticas y operaciones de carga/eliminación.
- Métodos:
    - `int buscar_pagina_id(int id)`
        - Objetivo: Buscar si una página está en el buffer.
        - Input: ID de página.
        - Output: índice o -1.

    - `int cargar_pagina(int id_bloque, Operacion op)`
        - Objetivo: Cargar página al buffer, aplicando LRU si necesario.
        - Input: ID de bloque, operación.
        - Output: índice del marco.

    - `int buscar_frame_libre()`
        - Objetivo: Buscar marco vacío.
        - Input: ninguno.
        - Output: índice o -1.

    - `int tarjet_eliminar()`
        - Objetivo: Elegir marco a eliminar por LRU.
        - Input: ninguno.
        - Output: índice del marco.

    - `void eliminar(int idx)`
        - Objetivo: Eliminar un marco y guardar su contenido si está sucio.
        - Input: índice del marco.
        - Output: ninguno.

    - `void guardar(int idx)`
        - Objetivo: Preguntar al usuario si desea guardar la página modificada.
        - Input: índice del marco.
        - Output: actualización del disco si se confirma.

    - `string* get_puntero(int idx)`
        - Objetivo: Obtener puntero al contenido de un marco.
        - Input: índice.
        - Output: puntero a string.

    - `void actualizar_tiempo_uso(int idx)`
        - Objetivo: Actualizar uso de marco para LRU.
        - Input: índice.

        - Output: ninguno.

    - `void incrementar_pin_count(int idx)` / `void incrementar_hit()` / `void incrementar_miss()`
        - Objetivo: Incrementar contadores de uso.
        - Input: índice (o ninguno).
        - Output: ninguno.

    - `void print()` / `void print_hit_rate()`
        - Objetivo: Mostrar estado de marcos y estadísticas.
        - Input: ninguno.
        - Output: impresión por consola.

    - `void pin(int id)` / `void unpin(int id)`
        - Objetivo: Marcar o desmarcar uso de marco.
        - Input: ID de página.
        - Output: ninguno.

    - `void high_dirty_bit(int id)`
        - Objetivo: Marcar una página como modificada.
        - Input: ID de página.
        - Output: ninguno.

Clase `Frame`

- Funcionalidad: Representa un marco en memoria, con su contenido (página) y metadatos.
- Métodos:
    - `Frame(int i)`
        - Objetivo: Inicializar el marco con ID e indicadores en falso.
        - Input: índice del marco.
        - Output: ninguno.

    - `void set_pagina(Page* p)`
        - Objetivo: Asociar una página al marco.
        - Input: puntero a página.
        - Output: ninguno.

    - `void reset_frame()`
        - Objetivo: Liberar la página y reiniciar atributos.
        - Input: ninguno.

        - Output: ninguno.

    - `int get_id()` / `int get_last_used()` / `bool get_is_pin()` / `bool get_dirty_bit()`
        - Objetivo: Obtener atributos del marco.
        - Input: ninguno.
        - Output: valor correspondiente.

    - `void pin()` / `void unpin()` / `void incrementar_pin_count()`
        - Objetivo: Marcar o desmarcar uso de marco.
        - Input: ninguno.
        - Output: ninguno.

    - `void set_last_used(int time)`
        - Objetivo: Establecer último uso.
        - Input: valor entero.
        - Output: ninguno.

    - `void high_dirty_bit()`
        - Objetivo: Marcar el marco como sucio.
        - Input: ninguno.
        - Output: ninguno.

    - `string* get_puntero()`
        - Objetivo: Devolver puntero al contenido de la página.
        - Input: ninguno.
        - Output: puntero a string.

    - `void ver_atributos()`
        - Objetivo: Imprimir todos los atributos del marco.
        - Input: ninguno.
        - Output: impresión por consola.

Clase Page

- Funcionalidad: Representa una página de disco, cargando su contenido al ser instanciada.
- Métodos:
    - `Page(int id)`
        - Objetivo: Leer los sectores correspondientes del disco y cargar el contenido.

        - Input: ID de bloque.
        - Output: ninguno.

    - `int get_id()`
        - Objetivo: Devolver el ID de la página.
        - Input: ninguno.
        - Output: entero.

Clase `DiscoFisico`

- Funcionalidad: Simula un disco físico estructurado jerárquicamente en discos, superficies, pistas y sectores. Permite su creación, lectura, escritura, inserción, modificación y reporte.
- Métodos:
    - `DiscoFisico()`
        - Objetivo: Constructor por defecto. Inicializa los atributos.
        - Input: ninguno.
        - Output: ninguno.

    - `void crear(char* nombre, unsigned int discos, unsigned int pistas, unsigned int sectores, unsigned int tam, unsigned int bloque)`
        - Objetivo: Crea la estructura física del disco en el sistema de archivos.
        - Input: nombre del disco, número de discos, pistas, sectores, tamaño por sector, tamaño de bloque.
        - Output: ninguno.

    - `bool inicializar(char* nombre)`
        - Objetivo: Carga la configuración del disco desde su cabecera.
        - Input: nombre del disco.
        - Output: booleano (true si se inicializó correctamente).

    - `bool modificar(string str, unsigned int d, int cara, unsigned int p, unsigned int s)`
        - Objetivo: Reemplaza el contenido de un sector específico.
        - Input: cadena a escribir, coordenadas del sector (disco, cara, pista, sector).
        - Output: booleano.

    - `void reporte()`

        - Objetivo: Imprime información detallada del estado y configuración del disco.
        - Input: ninguno.
        - Output: ninguno.

    - `string leer(char* ruta)`
        - Objetivo: Lee el contenido de un sector a partir de su ruta.
        - Input: ruta en formato `"d/cara/p/s"`.
        - Output: string con el contenido del sector.

    - `string leer(unsigned int d, int cara, unsigned int p, unsigned int s)`
        - Objetivo: Lee un sector mediante coordenadas directas.
        - Input: coordenadas del sector.
        - Output: string con el contenido del sector.

    - `bool escribir(char* str, unsigned int d, int cara, unsigned int p, unsigned int s, char* nombre)`
        - Objetivo: Escribe una cadena en un sector específico.
        - Input: cadena a escribir, coordenadas del sector, nombre lógico.
        - Output: booleano.

    - `bool escribirBloque(char* str, unsigned int d, int cara, unsigned int p, unsigned int s, char* nombre, char modo, int* lista_tamanos)`
        - Objetivo: Escribe datos distribuidos en sectores de un bloque con formato fijo.
        - Input: datos, coordenadas, nombre, modo (‘F’), lista de tamaños por campo.
        - Output: booleano.

    - `bool encontrarSector(char* ruta, int id_bloque, int idx) const`
        - Objetivo: Calcula la ruta del sector `idx` dentro de un bloque dado.
        - Input: ruta (puntero para salida), id de bloque, índice del sector.
        - Output: booleano.

    - `bool actualizarCabeceraFija(char* ruta)`
        - Objetivo: Actualiza la cabecera del sector con la metainformación y espacios disponibles.
        - Input: ruta del sector.

        - Output: booleano.

    - `bool insertarBloque(char* linea, int id_bloque, char* nombre, char modo, int* lista_tamanos)`
        - Objetivo: Inserta una línea de datos distribuidos entre sectores de un bloque.
        - Input: datos, id de bloque, nombre, modo, lista de tamaños.
        - Output: booleano.

    - `bool insertarFijo(char* str, char* ruta, char* nombre, int* lista_tamanos)`
        - Objetivo: Inserta una línea en formato fijo dentro de un sector específico.
        - Input: datos, ruta del sector, nombre, lista de tamaños.
        - Output: booleano.

    - `bool insertar(char* str, int tam, char* ruta, char* nombre)`
        - Objetivo: Inserta datos en un sector si hay espacio disponible.
        - Input: cadena, tamaño, ruta, nombre.
        - Output: booleano.

    - `bool insertar(char* str, int tam, unsigned int d, int cara, unsigned int p, unsigned int s, char* nombre)`
        - Objetivo: Inserta datos recorriendo sectores hasta encontrar espacio.
        - Input: datos, tamaño, coordenadas, nombre.
        - Output: booleano.

    - `bool avanzar(unsigned int& di, int& cara, unsigned int& pi, unsigned int& se)`
        - Objetivo: Avanza a la siguiente posición válida del disco.
        - Input: coordenadas por referencia.
        - Output: booleano (false si no hay más sectores disponibles).

    - `void registrarRelacion(char* sector, char* nombre)`
        - Objetivo: Registra la relación entre un nombre lógico y un sector.
        - Input: ruta del sector, nombre lógico.
        - Output: ninguno.

    - `void reemplazar(int id_bloque, string* contenido, bool es_insercion, char* nombre)`
        - Objetivo: Reemplaza o inserta contenido en un bloque completo.
        - Input: id de bloque, puntero a contenido por sector, indicador de inserción, nombre.

- Output: ninguno.
